Predicción Salario Mayor a $50,000 USD al Año

Carlos Velasco, ITC A01708634, Tecnológico de Monterrey Campus Querétaro

Abstracto.- Este documento presenta la implementación y explicación sobre la determinación de salarios anuales mayores a $50,000 USD en Estados Unidos basado en datos del consenso.

I. Introducción

El ingreso anual es un indicador clave del bienestar económico de los individuos y las familias, especialmente en países como los Estados Unidos. Ganar $50,000 al año es un punto de referencia significativo que a menudo se utiliza para diferenciar entre diferentes niveles de estabilidad financiera.

Para muchas personas, ganar menos de $50,000 USD al año puede significar dificultades para mantenerse al día con los costos de vida, especialmente en áreas urbanas con altos gastos de vivienda. Por otro lado, superar este umbral puede representar un mayor acceso a oportunidades y una mejor calidad de vida.

El presente análisis tiene como objetivo explorar los factores demográficos y socioeconómicos que influyen en la probabilidad de que una persona gane más de $50,000 USD al año. Utilizando datos censales, se busca construir un modelo predictivo que permita entender mejor las características que diferencian a aquellos que superan este umbral de ingresos de aquellos que no lo hacen.

II. Proceso de Análisis

El análisis de datos comienza con la identificación y selección de las variables más relevantes para alcanzar el objetivo: predecir si el ingreso anual de una persona excede los $50,000 USD. Este paso es fundamental, ya que determina las características que influyen en la capacidad predictiva del modelo.

Paso 1: Definición de la Información Necesaria

El primer paso fue identificar qué información del conjunto de datos censales es esencial para la predicción. Se seleccionaron características clave que reflejan aspectos demográficos, laborales, y sociales, tales como la edad, clase de trabajo, nivel educativo, estado civil, ocupación, relación familiar, raza, sexo, horas trabajadas por semana, y el país de origen. Estas variables son determinantes en la estructura económica de los individuos y consideré que pueden influir significativamente en sus niveles de ingresos.

Paso 2: Preparación de los Datos

Una vez identificadas las características necesarias, el siguiente paso fue preparar los datos para su uso en el modelo predictivo. Esto incluyó la transformación de las variables categóricas en numéricas, esto para que mi modelo pueda interpretar los valores de mis features de una manera más clara y consistente.

etl > dictionaries.py > ...

```
workclass_dict = {
    'Private': 1, 'Self-emp-not-inc': 2, 'Self-emp-inc': 3, 'Federal-gov': 4,
    'Local-gov': 5, 'State-gov': 6, 'Without-pay': 7, 'Never-worked': 8
}
marital_status_dict = {
    'Married-civ-spouse': 1, 'Divorced': 2, 'Never-married': 3, 'Separated': 4,
    'Widowed': 5, 'Married-spouse-absent': 6, 'Married-AF-spouse': 7
}
occupation_dict = {
    'Tech-support': 1, 'Craft-repair': 2, 'Other-service': 3, 'Sales': 4,
    'Exec-managerial': 5, 'Prof-specialty': 6, 'Handlers-cleaners': 7,
    'Machine-op-inspct': 8, 'Adm-clerical': 9, 'Farming-fishing': 10,
    'Transport-moving': 11, 'Priv-house serv': 12, 'Protective-serv': 13,
    'Armed-Forces': 14
}
```

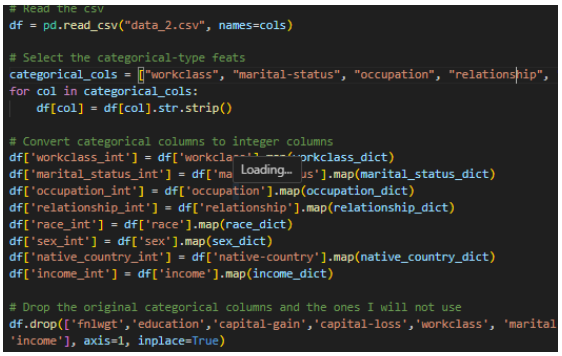
Figura 1. Transformación de variables categóricas en numéricas

Paso 3: Limpieza y Extracción de Datos

El proceso de limpieza de datos consistió en eliminar filas con valores faltantes (nulos) que podían comprometer la calidad del análisis. A pesar de la pérdida de un pequeño porcentaje de los datos (alrededor de un 7% de las instancias del dataset), esta decisión fue clave para asegurar la precisión y la fiabilidad del modelo. Después de la limpieza, los datos

fueron almacenados en un nuevo conjunto, listo para ser utilizado en los siguientes pasos del análisis.

Este proceso de análisis es crucial para garantizar que el modelo predictivo se base en datos sólidos y relevantes, lo que maximiza su capacidad para identificar patrones y tendencias que pueden influir en los ingresos de las personas.




```
# Read the csv
df = pd.read_csv("data_2.csv", names=cols)

# Select the categorical-type feats
categorical_cols = ["workclass", "marital-status", "occupation", "relationship",
for col in categorical_cols:
    df[col] = df[col].str.strip()

# Convert categorical columns to integer columns
df['workclass_int'] = df['workcla...orkclass_dict)
df['marital_status_int'] = df['ma...us'].map(marital_status_dict)
df['occupation_int'] = df['occupation'].map(occupation_dict)
df['relationship_int'] = df['relationship'].map(relationship_dict)
df['race_int'] = df['race'].map(race_dict)
df['sex_int'] = df['sex'].map(sex_dict)
df['native_country_int'] = df['native-country'].map(native_country_dict)
df['income_int'] = df['income'].map(income_dict)

# Drop the original categorical columns and the ones I will not use
df.drop(['fnlwgt','education','capital-gain','capital-loss','workclass', 'marital
'income'], axis=1, inplace=True)
```

Figura 2. Extracción de datos del set de datos y elección de columnas a utilizar en el modelo, así como la transformación de las mismas.

III. Set de Datos

El conjunto de datos utilizado en este análisis, conocido como "Census Income" o "Adult," es utilizado en estudios de clasificación dentro del campo de las ciencias sociales. Este dataset multivariado contiene 32,560 instancias y 14 características originales, aunque en el modelo final se utilizaron 30,019 instancias y una selección específica de features.

Características del Dataset
Área: Ciencias Sociales
Tareas Asociadas: Clasificación
Tipos de Features: Categóricas y Enteras
Características Seleccionadas para el Modelo Final:

- age (Edad)
- workclass (Clase de Trabajo)
- education-num (Nivel Educativo)
- marital-status (Estado Civil)
- occupation (Ocupación)
- relationship (Relación Familiar)
- race (Raza)
- sex (Sexo)
- hours-per-week (Horas Trabajadas por Semana)
- native-country (País de Origen)
- income (Ingreso)

Selección de Features
Las columnas seleccionadas para el modelo final fueron elegidas debido a su relevancia en la predicción de ingresos. Estas variables representan una combinación de factores demográficos, educativos, laborales y sociales que tienen un impacto directo en los niveles de ingresos de los individuos.

Es importante recalcar que las columnas 'fnlwgt', 'capital-gain', y 'capital-loss' no fueron consideradas debido a la falta de descripción y contexto suficiente para comprender sus características, valores y cómo éstas tenían relación directa con el objetivo del dataset. La columna 'education' fue omitida ya que sus valores eran redundantes al contar con la feature education-num.

Este dataset es un excelente recurso para analizar cómo diferentes aspectos de la vida de una persona pueden influir en su situación económica, proporcionando un marco sólido para la creación de modelos predictivos en el ámbito de las ciencias sociales.

IV. Propuesta de 1er Modelo

En esta sección, se presenta el modelo de regresión logística implementado para predecir si una persona tiene ingresos superiores a $50,000 anuales.

Regresión logística
Para dar un poco de contexto, la regresión logística es una técnica de modelado estadístico utilizada para predecir la probabilidad de un evento binario, es decir, un resultado que puede tener dos posibles valores (por ejemplo, sí/no, 0/1, verdadero/falso).

El modelo se basa en varios conceptos matemáticos y estadísticos clave, que se describen a continuación:

Función de Hipótesis: Sigmoide

La función sigmoide se utiliza como la función de hipótesis en nuestro modelo. Matemáticamente, se define como:

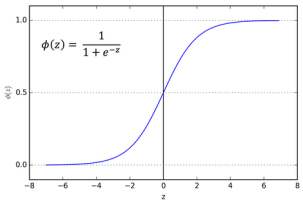


Figura 3. Función sigmoide

Esta función convierte cualquier valor real en un rango entre 0 y 1, lo que la hace ideal para modelos de clasificación binaria. En nuestro caso, $\sigma(z)$ representa la probabilidad de que una instancia pertenezca a la clase con ingreso >$50,000.

La sigmoide facilita la interpretación de las predicciones como probabilidades y es ideal para problemas de clasificación binaria.

Función de Pérdida: Entropía Cruzada

La entropía cruzada es la medida utilizada para calcular la pérdida o error del modelo. Esta métrica evalúa qué tan bien se están ajustando las predicciones a las etiquetas verdaderas. La fórmula general para la entropía cruzada es:

$$L(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right)$$

Figura 4. Función de pérdida entropía cruzada

En este modelo, se utiliza una versión ponderada de la entropía cruzada para dar más peso a las clases desbalanceadas (es decir, a las instancias con ingresos superiores a $50,000, que son menos comunes).

La entropía cruzada ponderada nos ayuda a tener una visualización sobre la capacidad del modelo para manejar conjuntos de datos desequilibrados, asegurando que las predicciones sean precisas y justas para ambas clases.

Optimización: Descenso de Gradiente

El descenso de gradiente es la técnica utilizada para optimizar los parámetros del modelo. Este método ajusta los parámetros para minimizar la función de pérdida. En cada iteración, los parámetros se actualizan en la dirección opuesta al gradiente de la función de pérdida con respecto a esos parámetros. La tasa de aprendizaje (learning rate, lr) controla el tamaño del paso que damos en cada iteración:

$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\Theta(x_i) - y) x_i]$$

Figura 5. Función de optimización de parámetros de descenso de gradiente.

Este proceso se repite hasta que la función de pérdida converge a un valor mínimo.

El descenso de gradiente ofrece un método eficiente para minimizar la función de pérdida, garantizando que el modelo aprenda de manera efectiva.

Normalización de Datos

En la construcción de modelos de machine learning, la normalización de los datos es un paso crucial, especialmente cuando los features tienen diferentes escalas. Al principio, no implementé la normalización en mi algoritmo, ya que no le veía mucho valor. Sin embargo, al analizar los resultados, noté que el algoritmo favorecía considerablemente las instancias con un target de 0 en comparación con las de 1. La mayoría de las predicciones resultaban en 0, lo que indicaba un sesgo en el modelo.

Al investigar más a fondo, identifiqué que este problema se debía a las columnas 'native-country' y 'hours-per-week', que contenían valores significativamente más altos

que otras columnas. Este desequilibrio en las escalas de los datos hizo que el algoritmo asignara más peso a las instancias con valores altos en estas columnas, lo que generaba predicciones sesgadas.

Para corregir este problema, implementé la función de normalización. La normalización transforma los datos para que todas las features tengan la misma escala, eliminando el sesgo inducido por las diferencias en magnitudes entre las columnas. La fórmula que utilicé para normalizar los datos es:

$$X_{\text{normalizado}} = \frac{X - \text{media}(X)}{\text{desviación estándar}(X)}$$

Figura 6. Función de normalización

Donde X representa los datos originales, media(X) es la media de cada feature y desviación estándar(X) es la desviación estándar de cada feature.

Esta transformación garantiza que cada feature tenga una media de 0 y una desviación estándar de 1, permitiendo que el algoritmo considere todas las features de manera equitativa. Después de aplicar esta normalización, el modelo dejó de favorecer instancias específicas y comenzó a hacer predicciones más equilibradas y precisas.

Primer Intento: Regresión Logística Deprecada

En el repositorio de GitHub, tengo un archivo llamado "logistic_reg_deprecated.py," que contiene mi primer intento de implementar la regresión logística. Este intento no funcionó del todo bien; aunque los parámetros se actualizaban correctamente, el proceso era extremadamente lento. El modelo tardó una hora en reducir el error a apenas 0.5.

El problema principal fue la implementación de batches. Un batch es un subconjunto del conjunto de datos completo utilizado para actualizar los parámetros en cada iteración. Sin embargo, este enfoque llevó a un overfitting, donde el error disminuía y luego aumentaba repetidamente. Esto indicaba que el modelo había aprendido patrones específicos de ciertos batches, lo que lo atrapaba en un bucle sin mejorar realmente su capacidad de generalización.

Algoritmo Optimizado

El nuevo algoritmo, "log_reg.py," incluye mejoras significativas sobre el primer intento. El uso de normalización y la eliminación de batches problemáticos han permitido que el modelo sea más eficiente y generalice mejor los patrones en el conjunto de datos. Ahora, el error converge de manera más consistente, lo que indica un modelo más robusto y confiable.

V. Train y Test

En machine learning, separar un dataset en conjuntos de entrenamiento (train) y prueba (test) es una práctica esencial para evaluar la capacidad de un modelo para generalizar a nuevos datos. Al entrenar un modelo únicamente en un subconjunto de los datos disponibles y luego probarlo en un conjunto separado, podemos estimar cómo funcionará el modelo en datos no vistos.

En mi algoritmo, implementé esta separación de la siguiente manera:

1. **Preparación de los Datos**:

   Primero, extraigo las features (x) y el target (y) del dataset limpio. El target, ‘income’, es la variable que queremos predecir.

2. **Shuffle de los Datos**:

   Antes de dividir el dataset en train y test, hago un shuffle (mezcla aleatoria) de los datos. Esto es crucial porque garantiza que la distribución de las instancias en los conjuntos de entrenamiento y prueba sea

representativa de la distribución total del dataset. Sin el shuffle, podríamos introducir un sesgo si, por ejemplo, las primeras filas del dataset estuvieran ordenadas de alguna manera.

3. **División en Train y Test**:

   Luego, divido los datos manualmente utilizando un 'train_ratio' del 80%, lo que significa que el 80% de los datos se utilizan para entrenar el modelo y el 20% restante para probarlo. Esta es una proporción comúnmente utilizada y proporciona un buen equilibrio entre la cantidad de datos para entrenar y evaluar el modelo.

4. **Normalización**:

   Después de dividir los datos en conjuntos de entrenamiento y prueba, aplico la normalización a ambos conjuntos. La normalización se realiza después de la división para evitar cualquier fuga de información entre el train y test, lo que podría conducir a un modelo con un rendimiento engañoso.

5. **Calibración del Modelo**:

   Calculé los pesos de las clases para manejar el desequilibrio en las clases, dado que los modelos pueden sesgarse hacia la clase mayoritaria, que en este caso, tengo más instancias con un target de 0 que de 1 (en mis cálculos es un ratio de 3:1 favoreciendo a las instancias con target de 0). Luego, inicialicé los parámetros que se optimizarían durante el entrenamiento del modelo.

6. **Entrenamiento del Modelo**:

   Utilicé el algoritmo de Gradient Descent para ajustar los parámetros del modelo durante un número fijo de iteraciones (epochs). A medida que el modelo se entrena, monitorizo el error usando la función de pérdida de cross-entropy ponderada.

7. **Predicciones**:

   Después del entrenamiento, realizo predicciones tanto en el conjunto de entrenamiento (train) como en el de prueba (test). Establecí un límite de 0.38 para clasificar las predicciones como 0 o 1, esto debido a que la mayoría de las predicciones estarán abajo de 0.5 porque hay el triple de instancias con target de 0 que de 1.

La división en conjuntos de entrenamiento y prueba, combinada con una evaluación cuidadosa de los resultados en ambos, asegura que el modelo tenga la capacidad de generalizar bien y no solo memorizar patrones específicos de los datos de entrenamiento (algo que me sucedió en mi primer implementación). Esto es fundamental para desarrollar modelos de machine learning robustos y confiables.

VI. Resultados del 1er Modelo

En esta sección, introduzco algunos elementos visuales que ayudan a interpretar los resultados de mi algoritmo de regresión logística.

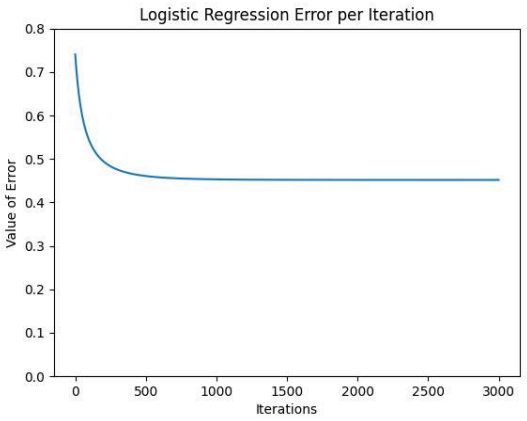


Figura 7. Error del modelo por iteración

La gráfica de error por iteración muestra cómo la pérdida de entropía cruzada ponderada

disminuye a lo largo de las iteraciones durante el entrenamiento del modelo de regresión logística.

La gráfica indica que el modelo ha alcanzado la convergencia, lo que significa que ha encontrado un conjunto de parámetros que minimizan la pérdida de manera efectiva.

A pesar de la estabilización, es importante seguir evaluando el rendimiento del modelo en los datos de prueba para asegurarse de que no esté sobreajustado, en otras palabras, que no esté haciendo un overfitting, que es justamente lo que se muestra en las siguientes gráficas.

Figura 8. Matriz de confusión para datos de entrenamiento

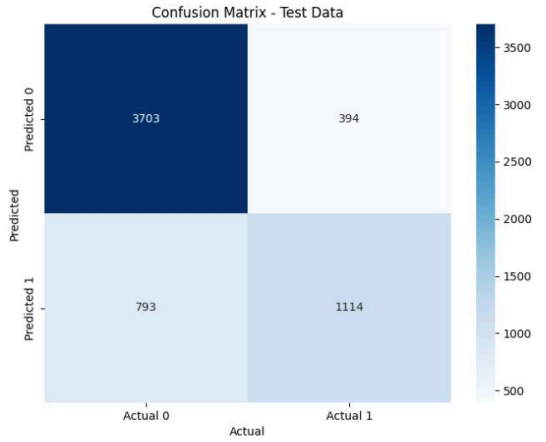


Figura 9. Matriz de confusión para datos de prueba

Las dos matrices de confusión anteriores representan la cantidad de predicciones correctas e incorrectas en mis datos de entrenamiento y de prueba.

Comparando las matrices de confusión de los datos de entrenamiento y prueba, podemos observar que el rendimiento del modelo es consistente en ambos conjuntos de datos. Esto sugiere que el modelo está generalizando bien y no está haciendo overfitting.

A continuación, daré mi interpretación a cada una de las posibles categorías que a mi algoritmo se le pueden asignar:

1. **Overfitting:** No hay una gran discrepancia entre las métricas de entrenamiento y prueba, lo que indica que el modelo no está sobreajustado.

2. **Underfitting:** El modelo está capturando bien las relaciones en los datos, ya que las métricas son razonablemente buenas en ambos conjuntos de datos. Esto fue gracias a los pesos que le asigné a ambas clases en el entrenamiento del modelo y al umbral que definí para determinar si una predicción es clase 0 o 1, aunque tengo que admitir que todavía hay espacio de mejora, en especial para la clase 1.

3. **Fitting**: El rendimiento en los datos de prueba es similar al de los datos de entrenamiento, lo que sugiere que el modelo está ajustando bien y generalizando adecuadamente.

VII. Conclusiones del 1er Modelo

Para finalizar, he llegado a la conclusión que el modelo está ajustando bien (fitting) debido a que las métricas de rendimiento en los datos de prueba son similares a las de los datos de entrenamiento, lo que indica que el modelo ha aprendido las relaciones subyacentes en los datos sin sobre ajustarse ni sub ajustarse. La gráfica de error por iteración también respalda esta conclusión, mostrando una disminución

rápida y estabilización del error, lo que sugiere que el modelo ha alcanzado la convergencia de manera efectiva.

Además, la consistencia en el rendimiento del modelo entre los conjuntos de datos de entrenamiento y prueba refuerza la idea de que los parámetros de mi modelo han sido sintonizados adecuadamente el cómo se relacionan los conjuntos de datos. Esta estabilidad dice que el modelo no solo ha aprendido correctamente los patrones presentes en el conjunto de entrenamiento, sino que también tiene la capacidad de generalizar esos patrones a datos no vistos previamente, algo que es sumamente valioso ya que sugiere que el modelo puede hacer predicciones acertadas con información nueva.

Sin embargo, es importante considerar que, aunque el modelo está mostrando un buen rendimiento, siempre existe la posibilidad de mejorar. Por ejemplo, ajustar más finamente los pesos asignados a las clases o explorar diferentes umbrales de clasificación podría aumentar la precisión, especialmente en la clase minoritaria, que en este caso es la clase 1. Asimismo, probar diferentes técnicas de regularización o realizar un análisis más exhaustivo de la matriz de confusión podría ayudar a identificar áreas de mejora para asegurar un rendimiento aún más robusto y equilibrado.

En resumen, los resultados obtenidos hasta el momento sugieren que el modelo está bien calibrado y tengo que decir que es un trabajo del que me siento muy orgulloso, aún estando consciente de sus áreas de oportunidad.

VIII. Mejorando el primer modelo

Conforme fui reforzando los conceptos de regresión logística, me di cuenta de ciertos aspectos que todavía no estaban presentes en mi modelo y/o interpretación:

- La definición del umbral para clasificar las clases en 0 o 1 no estaba basada en datos, simplemente en lo que pensaba que era mejor.
- El nivel de ajuste no estaba justificado en datos.
- Los hiper parámetros se ajustan hasta la capa de prueba
- No tenía en consideración el término de la multicolinealidad y los efectos negativos que esta puede tener en las estimaciones de los coeficientes del modelo final.

Teniendo en cuenta lo anterior, he desarrollado un segundo modelo en el cual pueda mejorar los diagnósticos de mi primer modelo.

IX. Propuesta del 2do Modelo

Utilizando mi primer modelo como base, implementé algunas técnicas para darle una mayor interpretación y justificación al algoritmo desarrollado.

En primer lugar, decidí implementar la curva ROC y calcular el AUC-ROC que me ayudarán a determinar cuál es el mejor umbral para la determinación de clases.

Curva ROC

La curva ROC es una herramienta utilizada para evaluar el rendimiento de un modelo de clasificación binaria. Representa la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) a medida que se varía el umbral de decisión del modelo.

AUC-ROC

El AUC-ROC es un valor numérico que resume la curva ROC en un solo número. Representa el área bajo la curva ROC y varía entre 0 y 1.

Un AUC-ROC cercano a 1 indica un modelo con un buen rendimiento de clasificación, mientras que un valor cercano a 0.5 sugiere un rendimiento similar al azar.

Estas dos herramientas me ayudan a tener una representación visual de cuál es la mejor tasa de verdaderos positivos en comparación a los falsos positivos. Encontrar este balance es lo

que me dará el umbral que necesito para determinar si las instancias que quiero predecir son de clase 0 (que indica que la persona gana menos de $50,000 USD anuales) o de clase 1 (que la persona gana igual o más de $50,000 USD anuales).

Asimismo, la curva ROC y el AUC-ROC me permiten visualizar el nivel de ajuste de mi modelo ya que, como se dijo anteriormente, una curva y un AUC-ROC cercanos a 1 indica un excelente nivel de rendimiento de clasificación, mientras que un valor más cercano a 0.5 indica que mi modelo es igual de bueno que un juego de azar.

Train, Test… y Validation

Adicionalmente, he agregado una nueva capa a la separación de mi conjunto de datos. Ahora, en lugar de ser solamente train y test, he añadido una capa intermedia llamada validación. Esta capa me ayuda a ajustar los hiper parámetros del modelo y evaluar su rendimiento durante el entrenamiento. También, el conjunto de prueba queda únicamente para evaluar el modelo y obtener una estimación realista de su rendimiento en datos no vistos.

Esta nueva separación de datos me permite separar el probar con diferentes combinaciones de hiper parámetros hasta tener diagnósticos adecuados (validación) y el obtener una comprensión de cómo se comportará el modelo en situaciones del mundo real (prueba).

Para crear este nuevo conjunto, tuve que hacer la división de los datos dentro del mismo train. Por lo tanto, de mi dataset original así quedaron distribuidos los conjuntos de datos:

1. Train: 64%
2. Validation: 16%
3. Test: 20%

Métodos de regularización

Finalmente, la última mejora que le agregué a mi modelo es la implementación de L2 Ridge.

La regularización L2 Ridge es una técnica utilizada en aprendizaje automático para prevenir el overfitting en modelos de regresión y clasificación. Funciona añadiendo un término de penalización a la función de costo que es proporcional a la suma de los cuadrados de los coeficientes del modelo.
Esto me ayuda a evitar que mi modelo memorice mi dataset y, por lo tanto, en el mundo real no sepa qué hacer cuando se le presentan datos que no ha visto.

También, esto dicta que las variables independientes tengan coeficientes adecuados y no se vean afectados por la multicolinealidad (que es cuando dos o más variables independientes tienen una fuerte correlación y se pelean unas contra otras para tener más influencia en las predicciones finales).

Todo esto sucede en el entrenamiento del modelo, entonces, la regularización L2 Ridge se ve reflejada en la función de gradiente descendiente (que es la que utiliza el algoritmo para entrenar a mi modelo).

X. Resultados del 2do Modelo

En esta sección, presento gráficas para demostrar el rendimiento del modelo así como su interpretación para determinar:

- ¿Qué tanto favorece mi modelo a una clase? (sesgo)
- ¿Qué porcentaje de error tiene mi modelo? (varianza)
- ¿Qué tan bueno es mi modelo para predecir valores nunca antes vistos? (nivel de ajuste).

Daré inicio a mi análisis de los resultados presentando la curva ROC y el AUC-ROC de cada conjunto de los datos.

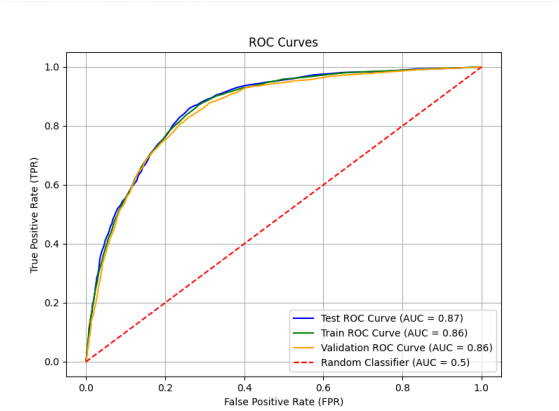


Figura 10. Curvas ROC y valores AUC-ROC para el 2do modelo

Como se puede observar en la gráfica, mis tres curvas (train, validation y test) están prácticamente sobrepuestas. Esto me quiere decir que el modelo tiene un rendimiento similar en términos de discriminación entre clases positivas y negativas (0 o 1) y, al tener un AUC-ROC de 0.87, sugiere que esta discriminación es bastante sólida.

En otras palabras, mi modelo generaliza bien en datos no vistos (conjunto de pruebas) y no está haciendo overfitting al conjunto de entrenamiento. Por otro lado, mi modelo no está haciendo underfitting debido a que los AUC-ROC están muy lejos de un valor de 0.5.

Asimismo, observamos que el valor de la AUC-ROC es prácticamente el mismo, lo que confirma la buena generalización de mi modelo en los 3 conjuntos.

Finalmente, la curva ROC me ayuda a determinar cuál es el mejor umbral para determinar si una clase es 0 o 1. Simplemente lo que tenemos que hacer es ubicar la gráfica en un punto donde consideremos que tenemos una buena tasa de verdaderos positivos contra falsos positivos.

En este caso, consideré que tener un 80% de precisión en predecir la clase 1 a costa de tener un 22% de error en predecir la clase 0 era un buen umbral para mi modelo. Esta es la clase de decisiones que nos permite hacer este tipo de herramientas y apoyos visuales.

Una vez encontrado ese umbral, podemos hacer las predicciones de nuestras clases y visualizar los resultados mediante nuestras matrices de confusión en cada uno de los 3 conjuntos de datos.

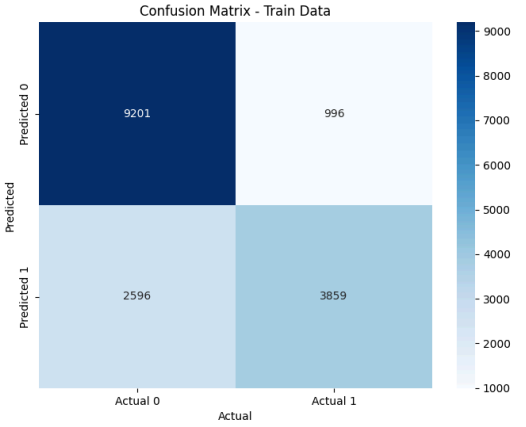


Figura 11. Matriz de confusión para el conjunto de entrenamiento para el 2do modelo

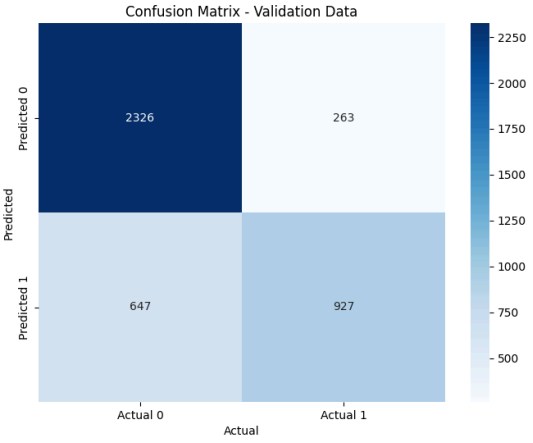


Figura 12. Matriz de confusión para el conjunto de validación para el 2do modelo

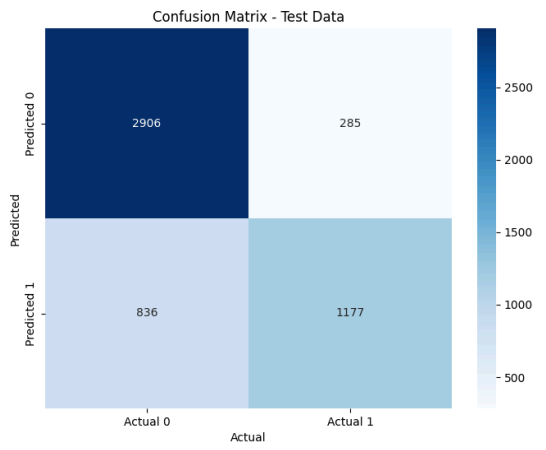


Figura 13. Matriz de confusión para el conjunto de prueba para el 2do modelo

Analizando a profundidad las gráficas, se observa que en los 3 conjuntos de datos se sigue un comportamiento similar.

Pero para no quedarnos en meramente intuición visual, vamos a calcular el error en cada una de las matrices para determinar qué tanto varía el error por clase en cada uno de los conjuntos de entrenamiento. Para ello llevaremos a cabo la siguiente fórmula:

Error clase 0: *1 - # de verdaderos negativos / # de verdaderos negativos + # de falsos positivos*
Error clase 1: *1 - # de verdaderos positivos / # de verdaderos positivos + # de falsos negativos*

Entrenamiento (train):
Error clase 0: 0.22
Error clase 1: 0.21

Validación (validation):
Error clase 0: 0.22
Error clase 1: 0.22

Prueba (test):
Error clase 0: 0.22
Error clase 1: 0.20

Desglosaré los resultados de la siguiente manera:

1. **Relación matriz de confusión y curva ROC:** Si prestamos atención, notaremos que estos resultados hacen total coherencia con la tasa seleccionada en la curva ROC (~80% de precisión en la clase 1 con ~22% de imprecisión en la clase 0).
2. **Sesgo:** El modelo no está siendo sesgado por ninguna de las dos clases de manera sustancial (está prediciendo un ~1% mejor la clase 1 que la clase 0), por lo tanto, podemos decir que en nuestro modelo existe un sesgo extremadamente bajo. Tomando en cuenta que hay un ratio de 3:1 en número de instancias que son clase 0 en el dataset original, considero que este es un increíble logro.
3. **Varianza:** El modelo tiende a equivocarse entre un 20% y 22% en ambas clases. Esto nos deja con una precisión del modelo de un 78% a 80% en cada uno de los conjuntos. Una diferencia del 2% entre las tasas de error entre mis 3 conjuntos de datos me da la justificación necesaria para determinar que la varianza es baja para mi modelo.
4. **Nivel de ajuste:** El modelo presenta grandes evidencias de una discriminación de clases fuerte (AUC-ROC de 0.87) y un comportamiento similar en los 3 conjuntos de datos (errores en las clases y curvas casi sobrepuestas en la curva ROC). Esto descarta las posibilidades de que mi modelo presente underfitting y overfitting, respectivamente. Por lo tanto, se puede determinar que mi modelo se está ajustando bien a datos no antes vistos y es eficiente en la determinación de clases. En otras palabras, mi modelo presenta fitting.

XI. Conclusiones del 2do Modelo

El segundo modelo desarrollado para predecir si una persona gana más de $50,000 USD al año ha demostrado un rendimiento sólido, basado en los resultados de las métricas clave evaluadas. Con un AUC-ROC de 0.87, el modelo exhibe una buena capacidad para discriminar entre personas que ganan más y menos de $50,000 USD, lo que indica que el modelo puede identificar con precisión a los individuos de la clase 1 (ingreso mayor a $50,000 USD) en la mayoría de los casos.

El hecho de que las curvas ROC estén casi alineadas de los conjuntos de entrenamiento, validación y prueba sugiere que el modelo generaliza bien a datos no vistos y no sufre problemas de overfitting ni underfitting. Además, la selección de un umbral de decisión que maximiza una precisión del 80% en la clase 1, a cambio de un error del 22% en la clase 0, muestra un balance razonable entre la predicción correcta de quienes ganan más de $50,000 USD y la tolerancia a errores en la clase de ingresos más bajos.

El análisis de la matriz de confusión confirma que el modelo tiene un sesgo mínimo entre ambas clases, manteniendo una diferencia de error de sólo 1% entre la clase 0 y la clase 1. Esto es un logro importante dado el desbalance de las clases en el conjunto de datos original, donde hay una proporción de 3:1 a favor de la clase 0. La baja varianza observada, con errores de entre 20% y 22% en ambos conjuntos, asegura que el modelo tiene un rendimiento consistente.

Si bien una tasa de error del 20-22% puede parecer considerable en ciertos dominios, como en el campo médico donde los errores pueden tener consecuencias críticas, en el contexto de predecir ingresos, el impacto no es tan grave. Sin embargo, sigue siendo importante, ya que predecir con precisión el nivel de ingresos tiene implicaciones en términos de identificar oportunidades económicas. Un error en esta predicción puede afectar la evaluación de las oportunidades o barreras financieras que enfrenta una persona, lo que podría influir en políticas o programas diseñados para mejorar el bienestar económico de ciertos grupos. Por lo tanto, aunque el margen de error es tolerable, hay margen para mejoras si se desea una mayor precisión en este aspecto.

En resumen, este modelo ofrece un buen nivel de predicción y puede ser un apoyo valioso para identificar factores críticos que contribuyen al bienestar económico, lo que facilita una mejor comprensión de la distribución de ingresos en los Estados Unidos.

XII. Solución con Framework

El uso de frameworks en la ciencia de datos y el aprendizaje automático es esencial debido a la facilidad y eficiencia que proporcionan en la implementación de modelos complejos. Frameworks como Scikit-learn, Pytorch, entre otros, ofrecen una amplia gama de algoritmos optimizados, facilitando tareas como la preprocesamiento de datos, la evaluación de modelos y la optimización de hiper parámetros. Utilizar estos frameworks permite:

- Reducción del tiempo de desarrollo, ya que muchas funciones necesarias para la construcción de modelos ya están implementadas y optimizadas.
- Mayor rendimiento en términos de velocidad de entrenamiento y predicción, debido a optimizaciones en el uso de recursos computacionales.
- Facilidad de experimentación con algoritmos avanzados sin necesidad de desarrollar cada detalle desde cero.

En mi caso, decidí emplear Scikit-learn para explorar distintas alternativas de solución, con el objetivo de mejorar el rendimiento y superar los problemas de overfitting observados en mi solución manual.

Estos fueron los diferentes acercamientos que hice para llegar a un mejor modelo que el manual que anteriormente analicé:

a) Regresión Logística con Framework
Inicialmente, intenté replicar mi solución manual utilizando Regresión Logística con el framework de Scikit-learn. Aunque este enfoque parecía prometedor al principio, experimenté overfitting significativo, lo que significa que el modelo ajustaba muy bien el conjunto de entrenamiento, pero su capacidad de generalizar a datos no vistos era limitada.

Específicamente, la clase 1 (aquellos que ganan más de $50,000 USD al año) no se predecía con la misma precisión que la clase 0, y cualquier intento de mejorar la predicción en la clase 1 llevaba a una disminución considerable en la precisión de la clase 0. A pesar de varios intentos de regularización y ajuste de hiper parámetros, la regresión logística no logró un buen equilibrio en la predicción de ambas clases sin incurrir en errores importantes.

b) Árbol de Decisión
Luego, decidí probar un Árbol de Decisión, que es un modelo que divide el espacio de características en regiones basadas en condiciones binarias sobre los valores de las características. Este enfoque me permitió visualizar de forma clara cómo se realizaban las decisiones para clasificar a los individuos.

Un Árbol de Decisión es un método popular de clasificación que funciona dividiendo los datos en subconjuntos más pequeños a medida que avanza por cada rama del árbol, hasta que llega a una predicción.

A pesar de sus ventajas en términos de interpretación, también experimenté overfitting, ya que el modelo se adaptaba demasiado a los datos de entrenamiento. Al igual que en la regresión logística, el árbol de decisión no lograba una buena generalización para datos no vistos.

c) Random Forest
Una vez identificado el problema de overfitting en los modelos anteriores, opté por un Random Forest, que es un conjunto de múltiples árboles de decisión entrenados con subconjuntos de datos aleatorios. Este enfoque permite suavizar las predicciones y reducir el overfitting característico de un solo árbol de decisión.

El Random Forest tiene la ventaja de ser más robusto, ya que combina las predicciones de múltiples árboles de decisión, lo que reduce la varianza y mejora la precisión en datos no vistos.

Con este enfoque, logré mejorar considerablemente los resultados. Sin embargo, aún observé una ligera diferencia en precisión entre el conjunto de entrenamiento (~91%) y los conjuntos de validación y prueba (~80%). A pesar de ajustar varios hiper parámetros como el número de árboles y la profundidad máxima, no conseguí eliminar por completo el overfitting.

d) XG Boost
Finalmente, implementé XG Boost, un algoritmo basado en boosting que entrena múltiples modelos secuenciales, donde cada uno intenta corregir los errores de su predecesor. Este enfoque me permitió reducir el overfitting que había experimentado con el Random Forest, aunque con un pequeño sacrificio en precisión en el modelo.

XG Boost es conocido por su capacidad de ofrecer altos niveles de rendimiento y precisión, siendo altamente eficiente en problemas con datos estructurados. A pesar de que la precisión en el conjunto de entrenamiento era un poco menor que en el Random Forest, los resultados en los conjuntos de validación y prueba eran mucho más consistentes, lo que indica que mi modelo generalizaba mejor a nuevos datos.

En la siguiente sección, presentaré una comparación detallada entre los resultados obtenidos con Random Forest y XG Boost, destacando por qué elegí XG Boost como mi solución final.

XIII. Resultados con Framework

Primeramente, vamos a revisar cómo se comportan las curvas ROC y qué valor nos da el AUC-ROC para el modelo de Random Forest.

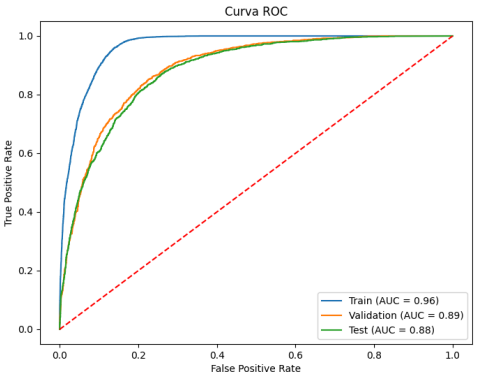


Figura 14. Curvas ROC y valores AUC-ROC para el modelo con framework (Random Forest)

Como podemos observar, el comportamiento de las curvas varía bastante de entrenamiento a validación y prueba, lo cual nos indica que el modelo no se ajusta bien a datos con los que no ha sido entrenado. Esto es malo debido a que en la vida real este va a ser el caso.

Igualmente, vemos la diferencia reflejada en los AUC-ROC, bajando de 0.96 a 0.88. Esta diferencia nos dice que el modelo es un poco peor para distinguir entre la clase positiva y negativa.

A continuación, discutiré los resultados obtenidos en la matriz de confusión para el modelo utilizando Random Forest. Es importante recalcar que para esta matriz de confusión los valores dentro de la matriz son diferentes, ya que también los calcule con un framework. En lugar de mostrar cuántas instancias pertenecen a positivos y negativos y cuántos son verdaderos o falsos, despliega el porcentaje de las instancias que mi modelo predijo (correcta o incorrectamente) si pertenecen a la clase 0 o 1.

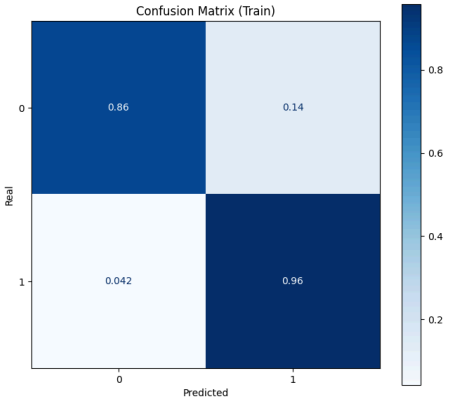


Figura 15. Matriz de confusión para el conjunto de entrenamiento para el modelo con framework (Random Forest)

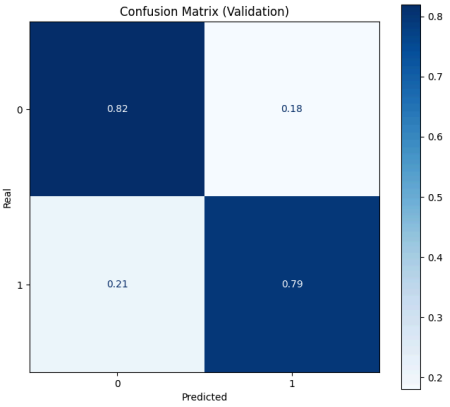


Figura 16. Matriz de confusión para el conjunto de validación para el modelo con framework (Random Forest)

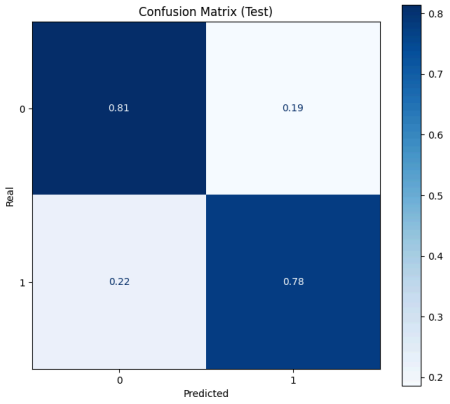


Figura 17. Matriz de confusión para el conjunto de prueba para el modelo con framework (Random Forest)

En la matriz de confusión de entrenamiento, vemos que tenemos un ligero sesgo a favor de la clase 1. Esto se debe a que en el entrenamiento del Random Forest le especifiqué que balanceara las clases y también limité la profundidad a 15. La razón por la que tomé esa decisión es porque:

- Si no especificaba el balanceo de las clases, entonces el sesgo hacia la clase 0 era evidentemente alto, lo que hacía que el modelo no podía predecir la clase 1 con efectividad.
- La profundidad de 15 limitaba a que mi modelo 'memorizara' los datos de entrenamiento de manera excesiva ya que, si incrementaba la profundidad, la efectividad de la predicción en training era casi perfecta, pero muy pobre en validación y prueba.

A pesar de esos ajustes, no pude evitar tener overfitting, y la varianza claramente se hace presente en de un conjunto de datos a otro.

Entrenamiento (train):
Error clase 0: 0.14
Error clase 1: 0.04

Validación (validation):
Error clase 0: 0.18
Error clase 1: 0.22

Prueba (test):
Error clase 0: 0.19
Error clase 1: 0.22

Es evidente que la varianza del error (en validation y test) es muy similar al de mi modelo hecho a mano y parte de mi objetivo de crear un modelo con framework es hacerlo más preciso y eficiente que el anterior.

Al final, los resultados de mi modelo con Random Forest se podrían categorizar de la siguiente manera:

- Nivel de ajuste: Overfitting
- Sesgo: Medio
- Varianza: Baja

Ahora, es tiempo de analizar los resultados obtenidos en nuestro modelo utilizando XG Boost.

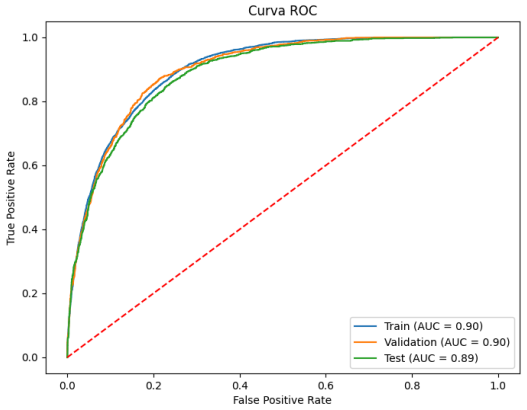


Figura 18. Curvas ROC y valores AUC-ROC para el modelo con framework (XG Boost)

En la curva ROC se observa que el modelo parece tener un buen poder predictivo, ya que todas las curvas están cerca de la esquina superior izquierda del gráfico, donde se desean altas tasas de verdaderos positivos y bajas tasas de falsos positivos. Esto se ve reflejado de la misma manera en nuestros valores AUC-ROC, donde los valores son considerablemente altos (0.90, 0.90 y 0.89), superando los valores que habíamos obtenido en nuestro modelo manual (0.87, 0.86 y 0.86) y sugiriendo que el modelo tiene una buena capacidad para distinguir entre las clases positiva y negativa.

En términos de niveles de ajuste, podemos observar que las curvas están muy cercanas la una de la otra, lo que sugiere que el modelo tiene un excelente ajuste con datos que no ha visto.

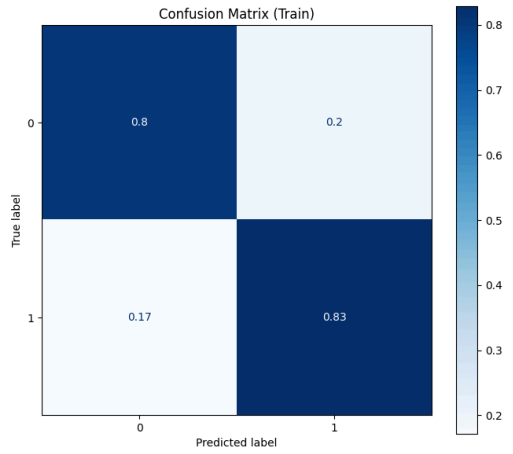


Figura 19. Matriz de confusión para el conjunto de entrenamiento para el modelo con framework (XG Boost)

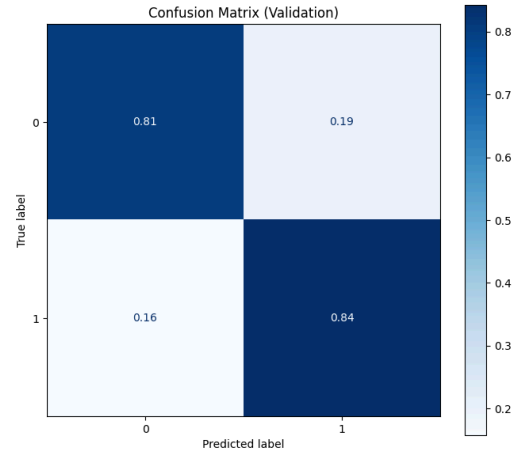


Figura 20. Matriz de confusión para el conjunto de validación para el modelo con framework (XG Boost)

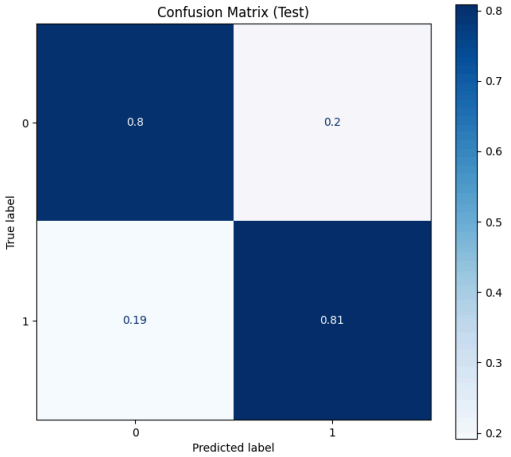


Figura 21. Matriz de confusión para el conjunto de prueba para el modelo con framework (XG Boost).

Antes de ir a la parte final de nuestros resultados, obtendremos los valores de los errores.

Entrenamiento (train):
Error clase 0: 0.2
Error clase 1: 0.17

Validación (validation):
Error clase 0: 0.19
Error clase 1: 0.16

Prueba (test):
Error clase 0: 0.2
Error clase 1: 0.19

Analizando las matrices de confusión, obtenemos que:

1. **Nivel de ajuste**: Las 3 matrices de confusión muestran tasas muy similares de falsos positivos y verdaderos positivos, lo que indica un nivel de ajuste muy bueno (fitting).
2. **Sesgo:** El sesgo parece bajo, ya que los errores en ambas clases son relativamente equilibrados.
3. **Varianza**: La varianza es baja, debido a que no hay grandes discrepancias entre las tasas de error de las matrices.

XIV. Conclusiones con Framework

Al comparar los resultados entre el modelo de Random Forest y XG Boost, podemos observar diferencias clave en el rendimiento y comportamiento de ambos.

Random Forest mostró un alto nivel de overfitting, con una diferencia notable entre los resultados en el conjunto de entrenamiento y los de validación/prueba.

El AUC-ROC pasó de 0.96 en entrenamiento a 0.88 en prueba, lo que indica que el modelo tuvo dificultades para generalizar bien en datos no vistos.

A pesar de los esfuerzos para balancear las clases y limitar la profundidad del modelo, no fue posible eliminar por completo el sobreajuste. Esto se refleja también en los errores por clase, donde el modelo logró un mejor rendimiento en la clase 1 durante el

entrenamiento, pero empeoró en la validación y prueba, especialmente en la predicción de la clase 1.

Por otro lado, XG Boost demostró ser una solución más robusta. Los resultados de las curvas ROC y los valores AUC-ROC fueron consistentemente altos en los tres conjuntos de datos (entrenamiento, validación y prueba), lo que sugiere un mejor poder predictivo y un modelo más generalizable para distinguir si una instancia es clase 0 o 1.

Además, las matrices de confusión reflejan un bajo sesgo y una baja varianza, con tasas de error similares en ambas clases y entre los diferentes conjuntos de datos.

En resumen, mientras que Random Forest logró buenos resultados en entrenamiento, XG Boost se destacó por su capacidad para generalizar mejor, lo que lo convierte en la mejor alternativa para este problema. Los ajustes en XG Boost permitieron reducir el overfitting, logrando un rendimiento más equilibrado y eficiente. Aunque los valores de precisión en entrenamiento fueron ligeramente menores que en Random Forest, el rendimiento en datos no vistos fue superior, lo que hace de XG Boost la opción más confiable para el modelo final.

XV. Bibliografía

¿Qué es la regresión logística? - Explicación del modelo de regresión logística - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/logistic-regression/

RandomForestClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

UCI Machine Learning Repository. (s. f.). https://archive.ics.uci.edu/dataset/2/adult

XGBoost Documentation — xgboost 2.1.1 documentation. (n.d.-b). https://xgboost.readthedocs.io/en/stable/